EXPRESS MAIL LABEL NO:
EV158254474US

SYSTEM AND METHOD FOR TRANSFERRING DATA IN STORAGE CONTROLLERS

Angel G. Perozo

Theodore C. White

William W. Dennin

[0001] CROSS REFERENCE TO RELATED APPLICATIONS:

[0002] This patent application claims priority to U.S. provisional patent application Serial Number: 60/444,339, Filed on 01/31/2003, Docket Number QE1048.USPROV, entitled "System and Method for Coalescing Data".

[0003] BACKGROUND OF THE INVENTION

[0004] 1. Field Of the Invention

[0005] The present invention relates generally storage device controllers, and more particularly to streamlining data flow in storage device controllers.

[0006] 2. Background

[0007] Conventional computer systems typically include several functional components. These components may include a central processing unit (CPU), main memory, input/output ("I/O") devices, and streaming storage devices (for example, tape drives)

(referred to herein as "storage device"). In conventional systems, the main memory is coupled to the CPU via a system bus or a local memory bus. The main memory is used to provide the CPU access to data and/or program information that is stored in main memory at execution time. Typically, the main memory is composed of random access memory (RAM) circuits. A computer system with the CPU and main memory is often referred to as a host system.

[0008] The storage device is coupled to the host system via a storage device controller that handles complex details of interfacing the storage devices to the host system. Communications between the host system and the controller is usually provided using one of a variety of standard I/O bus interfaces.

[0009] Conventionally, when data is read from a storage device, a host system sends a read command to the controller, which stores the read command into the buffer memory. Data is read from the device and stored in the buffer memory.

[0010] Typically when data enters the controller from an interface (for example, the "SCSI interface"), the data is MODN aligned (for example, MOD2]. The storage controller includes a buffer memory controller that moves data with a specific alignment, for example, a MOD4 alignment. Hence data must be padded such that it complies with the MOD4 alignment.

[0011] In addition, when data is moved from a buffer memory of the controller to the SCSI interface, it has to be re-aligned so that the SCSI interface can send the data out. For example, MOD4 aligned data must be re-aligned to MOD2 data so that it can be read from buffer memory.

[0012] Conventional controllers do not provide an efficient system for padding or removing the pad for efficiently transferring data.

[0013] Therefore, there is a need for a system to efficiently pad/remove the pad for moving data to/from a controller.

[0014] SUMMARY OF THE INVENTION

[0015] In one aspect of the present invention, a method for processing incoming data by a storage controller with a buffer controller coupled to a buffer memory is provided. The method includes, evaluating incoming data block size; determining if the incoming data requires padding; and padding incoming data such that the incoming data can be processed by the buffer controller. The incoming data after being padded may be stored in the buffer memory and the buffer controller pads incoming data in real time before being stored in the buffer memory.

[0016] In another aspect of the present invention, a method for reading data from the buffer memory

operationally coupled to the storage controller through the buffer controller is provided. The method includes determining if any pads need to be removed from the data; and removing pads from the data read from the buffer memory.

[0017] In another aspect of the present invention, a storage controller is provided. The storage controller includes a buffer controller that can be set in a mode to receive any MOD size data and includes a first channel with a FIFO for receiving incoming data via a first interface, wherein the incoming data is padded so that it can be stored in a buffer memory. Also, padding may be removed from any data that is read from the buffer memory and the buffer controller mode for receiving incoming data can be set by firmware.

[0018] In one aspect of the present invention, a controller can process any MOD size data by padding (or removing the pad). This allows the controller to be flexible and hence more useful in the fast changing storage arena.

[0019] BRIEF DESCRIPTION OF THE DRAWINGS

[0020] The foregoing features and other features of the present invention will now be described with reference to the drawings of a preferred embodiment. In the drawings, the same components have the same reference numerals. The illustrated embodiment is

intended to illustrate, but not to limit the invention. The drawings include the following Figures:

[0021] Figure 1A shows a block diagram of a controller, according to one aspect of the present invention;

[0022] Figure 1B shows a block diagram of a buffer controller, according to one aspect of the present invention;

[0023] Figure 2 shows a block diagram of Channel 1, according to one aspect of the present invention;

[0024] Figure 3 shows a block diagram showing data coming from an interface, according to one aspect of the present invention;

[0025] Figure 4 shows an example of data moving from buffer memory to an interface, according to one aspect of the present invention;

[0026] Figure 5 shows a flow diagram for padding data, according to one aspect of the present invention; and

[0027] Figure 6 shows a flow diagram for removing padding, according to one aspect of the present invention.

[0028] DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0029] To facilitate an understanding of the preferred embodiment, the general architecture and operation of a controller will initially be described. The specific architecture and operation of the

preferred embodiment will then be described with reference to the general architecture.

[0030] System of Figure 1A is an example of a streaming storage drive system (e.g., tape drive), included (or coupled to) in a computer system. The host computer (not shown) and storage device 115 communicate via port 102, which is connected to a data bus (not shown). In an alternate embodiment (not shown), the storage device 115 is an external storage device, which is connected to the host computer via a data bus. The data bus, for example, is a bus in accordance with a Small Computer System Interface (SCSI) specification. Those skilled in the art will appreciate that other communication buses known in the art can be used to transfer data between the drive and the host system.

[0031] As shown in Figure 1A, the system includes controller 101, which is coupled to SCSI port 102, port 114, buffer memory 111 and microprocessor 100. Interface 118 serves to couple microprocessor bus 107 to microprocessor 100. A read only memory ("ROM") omitted from the drawing is used to store firmware code executed by microprocessor 100. Port 114 couples controller 101 to device 115.

[0032] Controller 101 can be an integrated circuit (IC) that comprises of various functional modules, which provide for the writing and reading of data

stored on storage device 115. Microprocessor 100 is coupled to controller 101 via interface 118 to facilitate transfer of data, address, timing and control information. Buffer memory 111 is coupled to controller 101 via ports to facilitate transfer of data, timing and address information.

[0033] Data flow controller 116 is connected to microprocessor bus 107 and to buffer controller 108. A DMA interface 112 is connected to microprocessor bus 107. DMA Interface 112 is also coupled to data and control port 113 and to data bus 107.

[0034] SCSI controller 105 includes programmable registers and state machine sequencers that interface with SCSI port 102 on one side and to a fast, buffered direct memory access (DMA) channel on the other side.

[0035] Sequencer 106 supports customized SCSI sequences, for example, by means of a 256-location instruction memory that allows users to customize command automation features. Sequencer 106 is organized in accordance with the Harvard architecture, which has separate instruction and data memories. Sequencer 106 includes, for example, a 32-byte register file, a multi-level deep stack, an integer algorithmic logic unit (ALU) and other special purpose modules. Sequencer 106 support's firmware and hardware interrupts schemes. The firmware interrupt allows microprocessor 100 to initiate an operation within

Sequencer 106 without stopping sequencer operation. Hardware interrupt comes directly from SCSI controller 105.

[0036] Buffer controller (may also referred to as "BC") 108 connects buffer memory 111 to DMA I/F 112, a SCSI channel of SCSI controller 105 and to micro-controller bus 107. Buffer controller 108 regulates data movement into and out of buffer memory 111.

[0037] To read data from device 115, a host system sends a read command to controller 101, which stores the read, commands in buffer memory 111. Microprocessor 100 then read the command out of buffer memory 111 and initializes the various functional blocks of controller 101. Data is read from device 115 and is passed through DMA I/F 112 to buffer controller 108.

[0038] Figure 1B shows a block diagram of BC 108 with Channel 1 108A and Channel 0 108D. BC 108 also includes registers 108E and an Arbiter 108C. Arbiter 108C arbitrates channel 0 108D and channel 1 108A access to controller 108B. Register 108E is used for to store status information and assists in generating interrupts.

[0039] Figure 2 shows a block diagram, of Channel 1 108A. Channel 1 108A includes a FIFO 200 that receives data from SCSI interface 105. Channel 1 108A also includes a read assembly unit 201 and plural register

202, operationally coupled to a controller 203 that is coupled to a buffer memory 111.

[0040] Channel 1 register(s) 202 includes the following registers/counters that are used for padding data coming from SCSI interface 105 and removing the pads for data that is read from buffer 111:

(a) SCSI Block Size Register: This register holds data blocks destined for buffer 111.

(b) Channel 1 Block size register: This register holds a specific MOD size data block (for example, 512 bytes for MOD4 and 510 bytes for MOD2). This specifies the block size that is sent to buffer 111.

(c) Channel 1 Data Length Counter: This holds a MODN data block and counts how much data has been received. This register is loaded with the same value as the SCSI Block size register when the "Data Length Load Select" bit is reset, as described below. When the "Data Length Load Select" bit is set, Data Length Counter is loaded with Data Length Reload Register value.

(d) Channel 1 Data Length Reload Register: This register holds MODN data block size and is used to reload Data Length Counter.

(e) Channel 1 Control Register: This register includes the following bits used for controlling data transfer:

[0041] "Transfer Count Load Select" bit: This bit selects the transfer count (i.e. the block size).

[0042] "Assembly Mode" bit: This bit when set allows incoming data to be of any MOD size and padded for storage in buffer 111.

[0043] "Data Length Load Select" bit: This bit allows loading of data in Data Length Counter either from SCSI block Size register or Data Length Reload Register.

[0044] The foregoing bits may be set by controller 101 firmware such that any MOD size data can be processed (by setting the "Assembly Mode" bit).

[0045] It is noteworthy that the foregoing register configuration is shown to illustrate the adaptive aspects of the present and not to limit the present invention.

[0046] Figures 3 and 4 show block diagrams of how data is padded and padding is removed, according to one aspect of the present invention. Figure 3 shows incoming data 300 is received by FIFO 200. This data is received from SCSI interface 105 and may be MODN aligned, which may be different than how data is stored in buffer 111. Data 300 leaving FIFO 200 is padded (for

example MODN data is padded to MOD4) so that it can be stored in buffer 111.

[0047] For buffer 111 read operations, as shown in Figure 4, MOD4 data 401 enters FIFO 200 and then the pad is removed so that data can be read.

[0048] Figure 5 shows a flow diagram of process steps for padding incoming data. Turning in detail to Figure 5, in step S500, data (300) is received from SCSI interface 105 at FIFO 200.

[0049] In step S501, data block length is evaluated by using the data length counter in register 202.

[0050] In step S502, data from FIFO 200 is sent to buffer 111, after a first data length has expired, and padding requirement is ascertained. If no padding is required, data is sent directly to buffer 111.

[0051] If padding is required, then data is padded in step S503. For example, if data is received as MOD2 and buffer 111 data is to be stored as MOD4, then two bytes are added to change the data alignment from MOD2 to MOD4.

[0052] In step S504, padded data block(s) are sent until all the data has been transferred.

[0053] Figure 6 shows a flow diagram for removing padded information from data that is read from buffer 111. Turning in detail to Figure 6, in step S601, the padding block size is evaluated by Channel 1 controller 203. In step S601, the process starts data transfer

from buffer 111 and determines if any pad(s) need to be removed. If no pad(s) are to be removed, then in step S605, the process determines if the last block has been transferred. If the last block is not transferred, the process goes back to step S600. If the last block has been transferred, then the process stops in step S606.

[0054] If it is determined that pads have to be removed (in step S601), then in step S602, the pad(s) are removed. For example, as shown in Figure 4, MOD4 data is read from buffer 111 and after the pads are removed, MODN data is sent to SCSI interface 105.

[0055] In step S603, the process determines if the last block of data has been transferred from buffer 111. If the last block has not been transferred, then the process moves back to step S600, otherwise, the process stops at step S604.

[0056] In one aspect of the present invention, controller 101 can process any MOD size data by padding (or removing the pad). This allows controller 101 to be flexible and hence more useful in the fast changing storage arena.

[0057] Although the present invention has been described with reference to specific embodiments, these embodiments are illustrative only and not limiting. Many other applications and embodiments of the present invention will be apparent in light of this disclosure.